



RRRRRRRR	MM	MM	11	CCCCCCCC	000000	NN	NN	NN
RRRRRRRR	MM	MM	11	CCCCCCCC	000000	NN	NN	NN
RR RR	RR	MMMM	MMMM	1111	CC	00	00	NN
RR RR	RR	MMMM	MMMM	1111	CC	00	00	NN
RR RR	RR	MM	MM	11	CC	00	00	NNNN
RR RR	RR	MM	MM	11	CC	00	00	NNNN
RRRRRRRR	MM	MM	11	CC	00	00	NN	NN
RRRRRRRR	MM	MM	11	CC	00	00	NN	NN
RR RR	RR	MM	MM	11	CC	00	00	NNNN
RR RR	RR	MM	MM	11	CC	00	00	NNNN
RR RR	RR	MM	MM	11	CC	00	00	NN
RR RR	RR	MM	MM	111111	CCCCCCCC	000000	NN	NN
RR RR	RR	MM	MM	111111	CCCCCCCC	000000	NN	NN

LL		SSSSSSSS
LL		SSSSSSSS
LL		SS
LLLLLLLL		SSSSSSSS
LLLLLLLL		SSSSSSSS

(3) 113 DECLARATIONS  
(4) 161 RMS\$CONNECT1 - SEQUENTIAL-SPECIFIC CONNECT ROUTINE

0000 1 \$BEGIN RM1CONN,000,RM\$RMS1,<SEQUENTIAL AND COMMON CONNECT>  
0000 2  
0000 3  
0000 4 :\*\*\*\*\*  
0000 5 :\*  
0000 6 :\* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0000 7 :\* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0000 8 :\* ALL RIGHTS RESERVED.  
0000 9 :\*  
0000 10 :\* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0000 11 :\* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0000 12 :\* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0000 13 :\* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0000 14 :\* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0000 15 :\* TRANSFERRED.  
0000 16 :\*  
0000 17 :\* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0000 18 :\* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0000 19 :\* CORPORATION.  
0000 20 :\*  
0000 21 :\* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0000 22 :\* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0000 23 :\*  
0000 24 :\*  
0000 25 :\*\*\*\*\*  
0000 26 :\*

0000 28 :++  
0000 29 Facility: rms32  
0000 30  
0000 31 Abstract:  
0000 32 routine to perform sequential-specific  
0000 33 connect processing.  
0000 34  
0000 35 Environment:  
0000 36 star processor running starlet exec.  
0000 37  
0000 38 Author: l f laverdure, creation date: 5-JAN-1977  
0000 39  
0000 40 Modified By:  
0000 41  
0000 42 V03-023 SHZ0026 Stephen H. Zalewski, 04-May-1984  
0000 43 If we are creating a global buffer section, specify that  
0000 44 we do an expand region to make sure it comes out of P0 space.  
0000 45  
0000 46 V03-022 JEJ0025 J E Johnson 10-Apr-1984  
0000 47 Ensure that GBH and GBD lengths are quadword aligned.  
0000 48  
0000 49 V03-021 SHZ0011 Stephen H. Zalewski, 24-Feb-1984  
0000 50 Do not initialize TRC blocks when connecting with global  
0000 51 buffers. This was accidentally left on from SHZ0010.  
0000 52  
0000 53 V03-020 SHZ0010 Stephen H. Zalewski 06-Dec-1983  
0000 54 Fix the tracing code to work with multi-threaded RMS. This  
0000 55 is accomplished by using the interlock queue instructions.  
0000 56  
0000 57 V03-019 SHZ0009 Stephen H. Zalewski 19-Sep-1983  
0000 58 Replace call to RMSINIT\_SFSB with RMSINIT\_SFSB\_IRB. This is  
0000 59 to allow us to successfully stall using the irab.  
0000 60  
0000 61 V03-018 SHZ0008 Stephen H. Zalewski 10-Aug-1983  
0000 62 Bugcheck if we try to create a global buffers section  
0000 63 with global buffer count of zero.  
0000 64  
0000 65 V03-017 SHZ0007 Stephen H. Zalewski 28-Jul-1983  
0000 66 Implement cluster global buffers.  
0000 67  
0000 68 V03-016 SHZ0006 Stephen H. Zalewski 22-Jun-1983  
0000 69 Disable global buffers.  
0000 70  
0000 71 V03-015 SHZ0005 Stephen H. Zalewski 11-Apr-1983  
0000 72 Fix bug that caused a process to incorrectly map a global  
0000 73 buffer section.  
0000 74  
0000 75 V03-014 KPL0001 Peter Lieberwirth 23-Mar-1983  
0000 76 Fix v03-013 by reversing sense of branch.  
0000 77  
0000 78 V03-13 SHZ0004 Stephen H. Zalewski 21-Feb-1983  
0000 79 If XQP is being used, ignore any request for global buffers.  
0000 80 This is only a temporary restriction.  
0000 81  
0000 82 V03-012 LJA0055 Laurie J. Anderson 12-Jan-1983  
0000 83 Fill in MBF field in IRB with the value which is used  
0000 84

0000	85	:	V03-011 KBT0413	Keith B. Thompson	30-Nov-1982
0000	86	:	Change ifb\$w_devbufsiz to ifb\$l_devbufsiz		
0000	87				
0000	88	:	V03-010 SHZ0003	Stephen H. Zalewski,	14-Oct-1982 16:29
0000	89	:	Prevent a Sconnect from occurring if there is no device buffer		
0000	90	:	or the real-time device bit is set in the device characteristics		
0000	91	:	field (this is also a patch in 3.2).		
0000	92				
0000	93	:	V03-009 SHZ0002	Stephen H. Zalewski,	10-Sep-1982 16:43
0000	94	:	Remove anything and everthing to do with FRBs, SIFBs and SFDs		
0000	95	:	because they no longer exist.		
0000	96				
0000	97	:	V03-008 KBT0341	Keith B. Thompson	16-Sep-1982
0000	98	:	Don't allocate multiple gbsbs when multistreaming		
0000	99				
0000	100	:	V03-007 SHZ0001	Stephen H. Zalewski,	1-Sep-1982 15:29
0000	101	:	Modify so that global buffer section locking is now done		
0000	102	:	via the lock manger and the GBSB.		
0000	103				
0000	104	:	V03-006 KBT0299	Keith B. Thompson	24-Aug-1982
0000	105	:	Reorganize psects		
0000	106				
0000	107	:	V03-005 KDM0002	Kathleen D. Morse	28-Jun-1982
0000	108	:	Added \$PCBDEF.		
0000	109				
0000	110	--			
0000	111				

0000 113 .SBTTL DECLARATIONS  
0000 114  
0000 115  
0000 116 ; Include Files:  
0000 117 ;  
0000 118  
0000 119 ;  
0000 120 ; Macros:  
0000 121 ;  
0000 122  
0000 123 \$BDBDEF  
0000 124 \$BLBDEF  
0000 125 \$CCBDEF  
0000 126 \$DEVDEF  
0000 127 \$FABDEF  
0000 128 \$FIBDEF  
0000 129 \$FWADEF  
0000 130 \$GBBDEF  
0000 131 \$GBHDEF  
0000 132 \$GBSBDEF  
0000 133 \$IMPDEF  
0000 134 \$IRBDEF  
0000 135 \$IFBDEF  
0000 136 \$PCBDEF  
0000 137 \$PRVDEF  
0000 138 \$PSLDEF  
0000 139 \$RMSDEF  
0000 140 \$RABDEF  
0000 141 \$SECDEF  
0000 142 \$SSDEF  
0000 143 \$SFSBDEF  
0000 144 \$TRCDEF  
0000 145 \$WCBDEF  
0000 146  
0000 147 ; Equated Symbols:  
0000 148  
0000 149 ;  
0000 150  
0000 151 ROP=RAB\$L\_ROP\*8 ; bit offset to rop  
0000 152  
0000 153 ;  
0000 154 ; Own Storage:  
0000 155 ;  
0000 156  
0000 157 FAOCNTRL:  
0000 158 .ASCIC /\_RMSS!XL/ ; Control string to FAO for GS name.  
0000 159

4C 58 21 24 53 4D 52 5F 00' 08 0000 0009

0009 161 .SBTTL RMSCONNECT1 - SEQUENTIAL-SPECIFIC CONNECT ROUTINE  
0009 162  
0009 163 :++  
0009 164 RMSCONNECT  
0009 165  
0009 166 RMSCONNECT  
0009 167  
0009 168 this routine performs the following functions required  
0009 169 for connecting to sequential files:  
0009 170  
0009 171 1. perform various validity checks  
0009 172 2. if opened for block i/o allocate a lock bdb  
0009 173 3. allocate required bdb's and buffers and save count  
0009 174  
0009 175  
0009 176 Calling sequence:  
0009 177  
0009 178 entered via case branch from rm\$connect  
0009 179  
0009 180 Input Parameters:  
0009 181  
0009 182 ap argument list addr  
0009 183 r11 impure area addr  
0009 184 r10 ifab addr  
0009 185 r9 irab addr  
0009 186 r8 rab addr  
0009 187  
0009 188 Implicit Inputs:  
0009 189  
0009 190 the contents of the rab and irab blocks  
0009 191  
0009 192 Output Parameters:  
0009 193  
0009 194 r0 status  
0009 195  
0009 196 Implicit Outputs:  
0009 197  
0009 198 sets various fields in the irab and ifab.  
0009 199  
0009 200 Completion Codes:  
0009 201  
0009 202 the standard rms status code is set into r0 and  
0009 203 return is made to the user (not caller).  
0009 204  
0009 205 if any errors, all irab-related internal structures  
0009 206 are deallocated.  
0009 207  
0009 208 Side Effects:  
0009 209  
0009 210 none  
0009 211  
0009 212 note:  
0009 213  
0009 214 only 1 connected irab is allowed on a sequential file.  
0009 215 this routine assumes that this is the first irab in  
0009 216 ifab's irab chain.  
0009 217

0009 218 ;--  
0009 219  
0009 220 RMSCONNECT1::  
0009 221  
0009 222 ;  
0009 223 ; if open or create was done with bro specified (mixed block & record i/o),  
0009 224 ; check the bio rop bit and if set switch to block i/o only  
0009 225 ;  
0009 226  
0C 22 06 E1 0009 227 BBC #FAB\$V BRO,-  
AA 20 8A 000B 228 IFBSB\_FAC(R10),8\$ ; branch if bro not set  
22 2B E1 0010 229 BICB2 #FAB\$M BIO,-  
04 68 AA 0012 230 IFBSB\_FAC(R10)  
20 88 0016 231 BBC #RAB\$V BIO+ROP,(R8),8\$ ; indicate not limited to block i/o  
22 AA 0018 232 BISB2 #FAB\$M BIO,-  
001A 233 IFBSB\_FAC(R10) ; branch if bio clear in rop  
0C 6A 3E E1 001A 234 8\$: BBC #IFBSV DAP, (R10), 20\$ ; switch to block i/o only  
00000000'EF 16 001E 235 JSB NTSCONNECT  
03 50 E8 0024 236 BLBS R0, 20\$ ; do network connect  
000C 31 0027 237 BRW CLN1 ; continue on success  
22 AA 05 E1 002A 239 20\$: BBC #IFBSV\_BIO,IFBSB\_FAC(R10),-  
1B 6C 11 002E 240 CHKMBC ; branch if not block i/o  
6C 11 002F 242 BRB ALLOC ; go to alloc if block i/o

0031 244  
0031 245 :  
0031 246 : error processing  
0031 247 :  
0031 248 : record format undefined and doing record i/o processing  
0031 249 :  
0031 250 :  
0031 251 ERRRFM:  
0031 252 RMSCONN\_ERRRFM::  
FFC7: 30 0036 253 RMSERR RFM  
FFC4: 31 0039 254 CLN1: BSBW RMSCLN1  
003C 255 BRW RMSEX\_NOSTR ; deallocate irab  
003C 256 ; and exit  
003C 257 :  
003C 258 : disk buffer size not 512, device is realtime device, or device has a zero  
003C 259 : length device buffer.  
003C 260 :  
003C 261 :  
F3 11 003C 262 ERRDEV: RMSERR DEV  
0041 263 BRB CLN1  
0043 264 :  
0043 265 :  
0043 266 : mbc negative. (reserved for later use)  
0043 267 :  
0043 268 :  
0043 269 ERRMBC:  
EC 11 0048 270 RMSERR MBC  
004A 271 BRB CLN1  
004A 272

004A 274 .ENABL LSB

004A 275

004A 276

004A 277 ; allocate bdb\$ and i/o buffers of size = blocksize (\* mbc, if disk)

004A 278

004A 279 ;!!!!

004A 280 ;\note: might be nice to change rm\$aldbuf to do split-page

004A 281 ;(but not cross-page) allocations for unit record devices.\

004A 282 ;!!!!

004A 283 ;

004A 284 ;

004A 285 CHKMBC:

55 48 AA D0 004A 286 MOVL IFBSL\_DEVBUFSIZ(R10),R5 ; length of buffer

EC 13 004E 287 BEQL ERRDEV ; Cannot connect if no device buffer

1D E0 0050 288 BBS #DEV\$V RTM,- ; or device is realtime device.

E8 6A 0052 289 MOVL IFBSL\_PRIM\_DEV(R10),ERRDEV

54 01 D0 0054 290 #1,R4 ; mt offset for default mbf

0E E1 0057 291 BBC #DEV\$V FOD,-

4B 6A 0059 292 IFBSL\_PRIM\_DEV(R10),UNIT; branch if not disk or mt

18 E0 005B 293 BBC #DEV\$V FOR,-

47 6A 005D 294 IFBSL\_PRIM\_DEV(R10),UNIT; branch if mounted foreign

6A 1C E1 005F 295 BBC #DEV\$V\_RND,IFBSL\_PRIM\_DEV(R10),-

3A 0062 296 ALLOC ; branch if not disk

0063 297

0063 298 ; check that assumptions regarding disk buffer size are accurate

0063 299

0063 300

0063 301 ; otherwise some of sequential get code won't work

0063 302

0063 303

0200 8F 55 B1 0063 304 CMPW R5,#512

D2 12 0068 305 BNEQ ERRDEV ; it's all over if not that magic number!

006A 306

006A 307 ;

006A 308 ; this is a connect for a disk file.

006A 309

006A 310 ; process the mbc (multi-block count) field of the rab to determine

006A 311 ; the size of the buffers to be allocated.

006A 312

006A 313

50 37 54 D4 006A 314 CLRL R4 ; disk offset for default mbf

6A A8 98 006C 315 CVTBL RABSB\_MBC(R8),R0 ; get mbc

2E E1 0070 316 BBC #IFBSV\_PPF\_INPUT,(R10),-

08 0073 317 120\$

02 50 D1 0074 318 CMPL R0,#2

1A 1E 0077 319 BGEQU 130\$

50 02 D0 0079 320 MOVL #2,R0

15 12 007C 321 120\$: BNEQ 130\$

50 00000000'9F 98 007E 322 CVTBL #PIO\$GB\_DFMBC,R0

0C 12 0085 323 BNEQ 130\$

50 00000000'9F 98 0087 324 CVTBL #SYS\$GB\_DFMBC,R0

03 12 008E 325 BNEQ 130\$

50 01 D0 0090 326 MOVL #1,R0

AE 19 0093 327 130\$: BLSS ERRMBC

55 A9 50 01 83 0095 328 SUBB3 #1,R0,IRBSB\_MBC(R9)

55 50 A4 009A 329 MULW2 R0,R5

0071 30 009D 330 ALLOC: BSBW RM\$BDBALLOC

; go allocate the buffers

RM1CONN  
V04-000

SEQUENTIAL AND COMMON CONNECT  
RMSCONNECT1 - SEQUENTIAL-SPECIFIC CONNEC

L 7

16-SEP-1984 00:44:47 VAX/VMS Macro V04-00  
5-SEP-1984 16:23:11 [RMS.SRC]RM1CONN.MAR;1

Page 9  
(7)

1B 50 E8 00A0 331 150\$: BLBS  
FF5A' 31 00A3 332 BRW  
00A6 333 RM\$EX\_NOSTR

; continue on success  
; exit on error. error in  
; rm\$bdballoc returns everything

00A6 335 :  
 00A6 336 : buffer allocation for unit record and foreign mounted devices  
 00A6 337 :  
 00A6 338 : allocate a single buffer only  
 00A6 339 :  
 00A6 340 :  
 56 01 D0 00A6 341 UNIT: MOVL #1,R6 ; get just one buffer/bdb  
 02 E1 00A9 342 BBC #DEV\$V\_TRM,-  
 55 0200 0C 6A 00AB 343 IFBSL\_PRIM\_DEV(R10),160\$ ; go allocate if not term  
 05 1B 00AD 344 CMPW #512,R5 ; buffer size at least 512  
 55 0200 0F 8F 00B2 345 BLEQU 160\$ ; yes, use it  
 00A2 30 00B4 346 MOVW #512,R5 ; use 512 bytes as minimum  
 E2 11 00B9 347 160\$: BSBW RMSBDBALLOC\_ALT ; go allocate the buffer  
 00BC 348 BRB 150\$ ; do error check  
 00BE 349 .DSABL LSB  
 00BE 350 :  
 00BE 351 : perform remaining stream setup  
 00BE 352 :  
 00BE 353 :  
 00BE 354 :  
 3C A9 54 D0 00BE 355 SETNXT: MOVL R4,IRBSL\_NXTBDB(R9) ; set next bdb for seqxfr  
 00C2 356 :  
 00C2 357 :  
 00C2 358 : position file for stream at beginning of file  
 00C2 359 : unless eof bit set in ifab or rop  
 00C2 360 :  
 00C2 361 :  
 39 18 E0 00C2 362 BBS #DEV\$V\_FOR,-  
 6A 00C4 363 IFBSL\_PRIM\_DEV(R10),65\$ ; leave positioned at blk 0;  
 04 40 A9 D6 00C6 364 INCL IRBSL\_NRP\_VBN(R9) ; if non-file structured  
 6A 21 E0 00C9 365 BBS #IFBS\$V\_EOF,(R10),20\$ ; assume at beginning of file  
 OA 28 E1 00CD 366 BBC #RAB\$V\_EOF+ROP,(R8),30\$ ; branch if position to eof flag set  
 00D1 367 00D1 368 : branch if eof not set in rop either  
 00D1 369 :  
 00D1 370 : copy the eof position to the next record pointer context  
 00D1 371 :  
 00D1 372 :  
 40 A9 74 AA D0 00D1 373 20\$: MOVL IFBSL\_EBK(R10),IRBSL\_NRP\_VBN(R9); these better be zero  
 44 A9 5C AA B0 00D6 374 MOVW IFBSW\_FFB(R10),IRBSW\_NRP\_OFF(R9); for unit record devices  
 00DB 375 :  
 00DB 376 :  
 00DB 377 : check for positioned at or past eof unless unit record  
 00DB 378 :  
 00DB 379 :  
 00DB 380 30\$: ASSUME DEV\$V\_REC EQ 0  
 14 6A E8 00DB 381 BLBS IFBSL\_PRIM\_DEV(R10),50\$ ; branch if unit record  
 40 A9 D1 00DE 382 CMPL IRBSL\_NRP\_VBN(R9),-  
 74 AA 00E1 383 IFBSL\_EBK(R10) ; nrp past eof?  
 0D 1F 00E3 384 BLSSU 50\$ ; branch if not  
 07 1A 00E5 385 BGTRU 40\$ ; branch if definite yes  
 00E7 386 :  
 00E7 387 :  
 00E7 388 : nrp vbn = eof vbn  
 00E7 389 : must check byte in block to determine if at eof  
 00E7 390 :  
 00E7 391 :

5C AA 44 A9 B1 00E7 392  
04 1F 00EC 393  
00EE 394  
00F2 395 40\$: CMPW IRBSW\_NRP\_OFF(R9),IFBSW\_FFB(R10)  
04 1B 00F6 396 50\$: BLSSU 50\$ ; branch if np < eof  
00F8 397 CMPB #IRBSV\_EOF,(R9) ; set the eof flag  
FF01' 31 00FC 398 BLEQU 60\$ ; just 1 buffer allocated?  
00FF 400 SSB #IRBSV\_RAHWBH,(R9) ; branch if yes  
00FF 401 RMSEXSOC ; enable read ahead & write behind  
00FF 402 ; maintains eof context on foreign devices  
00FF 403 ;  
00FF 404 ;  
F9 6A 05 E1 00FF 405 65\$: BBC #DEV\$V\_SGD,IFBSL\_PRIM\_DEV(R10),60\$; branch if not magtape  
F5 6A 21 E1 0103 406 BBC #IFBSV\_EOF,(R10),60\$ ; if not at eof, no problem  
0107 407 SSB #IRBSV\_EOF,(R9) ; set irab eof bit  
EF 11 010B 408 BRB 60\$ ; return to mainline

```

010D 410
010D 411 :++
010D 412
010D 413 subroutine to allocate bdb's and buffers. code to lock buffers in working
010D 414 set remains no-op'd at time of release 2. it is felt at this time that
010D 415 locking buffers in the working set when the multi-buffer count is positive
010D 416 will probably cause problems with existing programs because in fact, rms
010D 417 has not been locking them at all. if this is implemented in a future
010D 418 release, the cleanest technique would seem to be the addition of yet
010D 419 another (sigh...) rop bit in the rab as input to the $connect operation
010D 420 to specifically request rms to lock the buffers. the current behavior
010D 421 of using absolute value of the mbf field or default counts will continue.
010D 422
010D 423 if this is for magtape with truncate access, only 1 buffer is allocated
010D 424
010D 425 entry at rm$bdballoc_alt is used when buffer size is already specified in
010D 426 r6. see additional comments there.
010D 427
010D 428
010D 429 inputs:
010D 430
010D 431 r11 impure area address
010D 432 r10 ifab address
010D 433 r9 irab address
010D 434 r8 rab address
010D 435 r5 size of buffers to allocate, in bytes.
010D 436 r4 index for defaults, values as follows:
010D 437
010D 438 0 - sequential disk file default
010D 439 1 - magtape default
010D 440 2 - unit record default
010D 441 3 - relative file default
010D 442 4 - indexed file default
010D 443 5 - hashed file default
010D 444
010D 445 rab$bb_mbf explicit # of buffers
010D 446
010D 447 outputs:
010D 448
010D 449 r0 status code
010D 450 r1-r6 destroyed
010D 451 r4 address of last bdb allocated
010D 452 irb$bb_bcnt # of buffers allocated - updated only if r9 nonzero.
010D 453
010D 454 allocation failure when called from connect (r9 nonzero) will
010D 455 return all allocated buffers, bdb's, bcb's, and the irab.
010D 456
010D 457 :--
010D 458
55 7C 010D 459 BLKALL: CLRQ R5 : this will get lock bdb only
4D 11 010F 460 BRB RMS$bdballoc_alt ; extended branch
22 AA 05 E0 0111 461 RMS$bdballoc::: ;-
F7 0115 462 BBS #IFB$V_BIO,IFBSB_FAC(R10),-
56 36 A8 98 0116 463 BLKALL ; block i/o then just do bdb
1F 12 011A 464 CVTBL RAB$B_MBF(R8),R6 ; get number of buffers
56 00000000'9F44 98 011C 465 BNEQ 10$ ; branch if specified
CVTBL @#PIO$GB_DFMBFSDK[R4],R6; else, pick up process default

```

56 00000000'9F44 15 12 0124 467 BNEQ 10\$ ; branch if specified  
                   0B 98 0126 468 CVTBL @#SYSSGB\_DFMBFSRK[R4],R6; else, pick up system default  
                   56 01 D0 012E 469 BNEQ 10\$ ; branch if specified  
                   0130 470 MOVL #1,R6 ; else use 1 buffer  
                   0133 471  
                   0133 472 ; if read ahead or write behind spec'd, then need two buffers  
                   0133 473 ;  
                   0133 474 ;  
                   0133 475 ;  
                   0133 476 ASSUME <<RAB\$M\_RAH!RAB\$M\_WBH>&^xFFFF00FF> EQ 0  
                   0133 477 BITW #<RAB\$M\_RAH!RAB\$M\_WBH>@-8,RAB\$B\_R0P1(R8)  
                   0137 478 ; either rab or wbh spec'd?  
                   0137 479  
                   02 13 0137 480 BEQL 10\$ ; eql don't want rah/wbh  
                   56 D6 0139 481 INCL R6 ; need min two buffs  
                   56 D5 013B 482 10\$: TSTL R6  
                   03 14 013D 483 BGTR 20\$ ; if pos, then ok  
                   56 56 CE 013F 484 MNEGL R6,R6 ; otherwise make it positive  
                   0142 485 20\$: MOVBL R6,IRB\$B\_MBF(R9) ; Save MBF value used  
                   5C A9 56 90 0142 486 BBC #IFB\$V\_TEF,(R10),40\$ ; branch if no truncate access  
                   07 6A 36 E1 0146 487 BBC #DEV\$V\_SQD,-  
                   05 E1 014A 488 IFB\$L\_PRIM\_DEV(R10),40\$ ; branch if not magtape  
                   03 6A 014C 489  
                   56 01 D0 014E 490 MOVL #1,R6 ; allocate 1 buffer  
                   0151 491 40\$:  
                   0151 492  
                   0151 493 ;  
                   0151 494 ; since we can't get good indexed defaults any other way  
                   0151 495 ; alter r6 here. indexed files require at least 2 bdb's and buffer's  
                   0151 496 ; so if absolute value of r6 is 1, then need to change it  
                   0151 497 ;  
                   0151 498 ;  
                   54 04 D1 0151 499 CMPL #4,R4 ; see if indexed  
                   08 12 0154 500 BNEQ 80\$ ; if not branch  
                   02 56 D1 0156 501 CMPL R6,#2 ; at least 2 buffers spec'd?  
                   03 1E 0159 502 BGEQU 80\$ ; ok if greater than or equal  
                   56 02 D0 015B 503 MOVL #2,R6 ; use 2 otherwise  
                   015E 504 80\$:  
                   015E 505  
                   015E 506 ;  
                   015E 507 ; alternate entry point for number of buffers already specified in r6.  
                   015E 508 ;  
                   015E 509 ; if r9 is zero, then irb\$b\_bcnt is not filled in. this entry point is  
                   015E 510 ; for unit record and foreign devices to allocate a single buffer not  
                   015E 511 ; using the mbf or defaults. extend and display will use this to allocate  
                   015E 512 ; buffers when no streams are connected (relative or isam only).  
                   015E 513 ;  
                   015E 514 ; inputs:  
                   015E 515 ;  
                   015E 516 ; r6 number of buffers to  
                   015E 517 ; allocate. 0 causes only one buffer to  
                   015E 518 ; be allocate and bypasses potential allocation  
                   015E 519 ; of lock bdb for relative and isam orgs.  
                   015E 520 ;  
                   015E 521 ; if set, then allocate a lock bdb also for  
                   015E 522 ; relative and isam files if low word r6 non zero  
                   015E 523 ;

015E 524 : ifb\$lsfsb\_ptr if non-zero, file is shared and a bcb is  
 015E 525 : allocated for each bdb.  
 015E 526 :  
 015E 527 : outputs:  
 015E 528 :  
 015E 529 : bdb's are linked into the end of the ifab bdb list.  
 015E 530 :  
 015E 531 :  
 015E 532 RMSBDBALLOC\_ALT::  
 04 22 AA 05 D4 015E 533 CLRC -(SP) : init buffer counter  
 06 6A 06 E1 0160 534 BBC #IFBSV\_BIO, IFBSB\_FAC(R10), AGAIN; Br if not block i/o.  
 0165 535 SSB #IFBSV\_NORECLK, (R10) ; Make sure noreclk is set for bio.  
 0169 536 AGAIN:  
 06 6A 06 E1 0169 537 INCL (SP) : count the buffer  
 FE92 30 016B 538 BSBW RM\$ALDBUF : allocate the buffer  
 23 50 E9 016E 539 BLBC RO,DECR BCNT : get out on error  
 0084 CA B6 017F 545 INCW IFBSW\_AVLCL(R10) : branch if no record locking.  
 E3 56 F5 0183 546 20\$: SOBGTR R6, AGAIN : Allocate a BLB.  
 0C 19 0186 547 BEQL 20\$ : branch if error on getting bcb  
 0188 548 BLSS DECR\_BCNT : was buffer allocated?  
 0188 549 : EQL then not, so don't count it.  
 0188 550 : note buffer allocated.  
 0188 551 : decrement counter, go again  
 0188 552 : if still positive  
 0188 553 : this was last pass to alloc  
 0188 554 : just lock bdb so decr bcnt  
 0188 555 : so it only counts buffers  
 0188 556 :  
 0188 557 : At this point the required number of buffers and bdbs, and blbs (if shared)  
 0188 558 : have been allocated. Allocate a lock blb if record locking is being done.  
 0188 559 :  
 0188 560 :  
 0A 6A 33 E0 0188 557 BBS #IFBSV\_NORECLK,(R10),EXIT : done if no locking.  
 FE71 30 018C 558 BSBW RM\$ALB[B : Allocate a lock BLB.  
 24 50 E8 018F 559 BLBS RO, CHKGL : Check out global buffers.  
 02 11 0192 560 BRB EXIT : Exit on error from alblb.  
 0194 561 DECR\_BCNT:  
 0194 562 :  
 0194 563 : come here on error and  
 0194 564 : last pass to get count right  
 0194 565 :  
 0194 566 :  
 0194 567 :  
 6E D7 0194 568 DECL (SP) :  
 0196 569 EXIT: POPR #^M<R1> : get buffer count off stack  
 02 BA 0196 570 TSTL R9 : is there an irab?  
 59 D5 0198 571 BEQL 10\$ : no, then don't update bcnt  
 07 13 019A 572 : and exit (caller checks error)  
 019C 573 : store count of buffers  
 54 A9 51 90 019C 574 MOVB R1,IRB\$B\_BCNT(R9) : error on allocation  
 01 50 E9 01A0 575 BLBC R0,20\$ : clean up buffers allocated  
 01A3 576 : and get rid of irab  
 01A3 577 : r9 nonzero means this was  
 01A3 578 : called on a connect  
 01A3 579 : and exit routine  
 05 01A3 580 10\$: RSB :  
 ;

50 DD 01A4 581 20\$: PUSHL R0 ; save status  
FE57' 31 01A6 582 BRW RM\$COMCLNUP ; and branch to cleanup  
01A9 583  
01A9 584 :  
01A9 585 : we couldn't get a blb for some reason (e.g., not enough space left).  
01A9 586 : therefore, we must return the bdb we just got.  
01A9 587 :  
01A9 588  
01A9 589 GIVEBACK:  
54 44 50 DD 01A9 590 PUSHL R0 ; save status code  
AA DO 01AB 591 MOVL IFBSL\_BDB\_BLNK(R10),R4 ; get back link because  
01AF 592 aldbuf calls albdb which  
01AF 593 links them at end of list  
FE4E' 30 01AF 594 BSBW RM\$RETBDB ; deallocates bdb ar4  
01 BA 01B2 595 POPR #^M<R0>  
DE 11 01B4 596 BRB DECR\_BCNT ; restore status code  
01B6 597 ; fix count and exit

01B6	599						
01B6	600	;					
01B6	601	;	Code to allocate global buffers if desired, and initialize if necessary.				
01B6	602	;					
01B6	603	;					
01B6	604	CHKGBL:					
78 AA	D5	01B6	605	TSTL	IFB\$L_SFSB_PTR(R10)	If file is not being shared,	
DB	13	01B9	606	BEQL	EXIT	then don't bother with global buffers	
0088 CA	D5	01BB	607	TSTL	IFB\$L_GBH_PTR(R10)	If we already have global buffers	
03	13	01BF	608	BEQL	1\$	then multi-streaming.	
00A5	31	01C1	609	BRW	MAP_IT	Go map the section.	
CF	6B	E9	01C4	610	ASSUME 1\$:		
CB	6B	05	E0	01C7	611 BLBC (R11), EXIT	Just use local if this is ppf.	
		59	D5	01CB	612 BBS #IMP\$V_NOP0BUFS,(R11),EXIT	Use local if P0 off limits.	
		C7	13	01CD	613 TSTL R9	Is irab present?	
56	3C	A8	D0	01CF	615 MOVL RAB\$L_FAB(R8), R6	No, then just use local.	
56	48	A6	32	01D3	616 CVTWL FABSW_GBC(R6), R6	Get address of FAB.	
		BD	13	01D7	617 BEQL EXIT	Get gbl buffer count.	
		FE24	30	01D9	618 BSBW RMSINIT_GBSB	None desired, so exit.	
		B7	50	E9	01DC	619 BLBC RO, EXIT	Allocate GBSB and get EX lock.
54	7C	AA	D0	01DF	620 MOVL IFB\$L_GBSB_PTR(R10), R4	Exit if lock failed.	
52	38	A4	D0	01E3	621 MOVL GBSB\$[GS_SIZE(R4), R2	Get address of GBSB from IFB.	
		54	13	01E7	622 BEQL CHK_GBC	Are global buffers already in use?	
		007D	31	01E9	623 BRW MAP_IT	No, branch to validate GBC.	
				01EC	624	Yes, go use them.	
54	009C	CA	D0	01EC	625 ERL0: MOVL IFB\$L_BLBBLNK(R10), R4	Get addr of BLB.	
		FE0C	30	01F1	626 BSBW RMSRETBLB	Return it.	
54	009C	CA	D0	01F4	627 ERL1: MOVL IFB\$L_BLBBLNK(R10), R4	Get BLB addr.	
		FE04	30	01F9	628 BSBW RMSRETBLB	Give back.	
54	44	AA	D0	01FC	629 ERL4: MOVL IFB\$L_BDB_BLNK(R10), R4	Get address of GBPB just alloc'd.	
		FDFD	30	0200	630 BSBW RMSRETGBP	Give it back.	
54	44	AA	D0	0203	631 ERL3: MOVL IFB\$L_BDB_BLNK(R10), R4	Get address of GBPB just alloc'd.	
		FDF6	30	0207	632 BSBW RMSRETGBP	Give it back.	
54	009C	CA	D0	020A	633 ERL2: MOVL IFB\$L_BLBBLNK(R10), R4	Get addr of a BLB (lock BLB).	
		FDEE	30	020F	634 BSBW RMSRETBLB	Give it back.	
		FDEB	30	0212	635 BSBW RMSRLS_GBSB	Dequeue the lock we had on the GBSB	
		01	BA	0215	636 POPR #^M<R05	Restore error code.	
		FF7C	31	0217	637 BRW EXIT	Go finish up.	
			021A	638	ALBLBERR:		
5E	08	C0	021A	640 ADDL2 #8, SP	Clean off stack.		
		50	DD	021D	641 PUSHL R0	Save error code.	
		D3	11	021F	642 BRB ERL1	Br and finish up.	
5E	08	C0	0221	643 ALBLBERR1:			
		50	DD	0224	644 ADDL2 #8, SP	Clean off stack.	
		D4	11	0226	645 PUSHL R0	Save error code.	
			0228	646 BRB ERL4	Br and finish up.		
5E	08	C0	0228	647 ALGBPERR:			
		50	DD	022B	648 ADDL2 #8, SP	Clean off stack.	
		DB	11	022D	649 PUSHL R0	Save error code.	
			022F	650 BRB ERL2	Br to give back lock BLB.		
5E	08	C0	022F	651 ALGBPERR1:			
		50	DD	0232	652 ADDL2 #8, SP	Clean off stack.	
		CD	11	0234	653 PUSHL R0	Save error code.	
			0236	654 BRB ERL3	Br to give back one gpb.		
			655 BAD_GBC:				

CD 11 0236 656 RMSERR GBC -(SP) ; Note error.  
 0238 657 BRB ERL2 ; Give back lock BLB.  
 023D 658  
 023D 659 ASSUME <>GBH\$C\_BLN/8>\*8> EQ GBH\$C\_BLN ; Check for quadword alignment  
 023D 660 ASSUME <>GBD\$C\_BLN/8>\*8> EQ GBD\$C\_BLN ; in GBD and GBH sections  
 023D 661  
 023D 662 **CHK\_GBC:**  
 52 56 D0 023D 663 MOVL R6, R2 ; Save number of buffers desired.  
 F4 19 0240 664 BLSS BAD\_GBC ; Only positive values allowed.  
 51 52 55 C5 0242 665 MULL3 R5, R2, R1 ; Total buffer bytes into R1.  
 52 28 C4 0246 666 MULL2 #GBD\$C\_BLN, R2 ; R2 is now descriptor bytes.  
 52 51 C0 0249 667 ADDL2 R1, R2 ; Sum of desc and buffers.  
 52 00000058 8F CO 024C 668 ADDL2 #GBH\$C\_BLN, R2 ; Plus size of header area.  
 52 000001FF 8F CO 0253 669 ADDL #511, R2 ; Round up to even pages.  
 52 000001FF 8F CA 025A 670 BICL #511, R2  
 06 11 0261 671 BRB MAP\_IT  
 00000004'EF 17 0263 672 JMP ADDTRC ; Noop to branch to ADDTRC for tracing.  
 14 BB 0269 673 **MAP\_IT:**  
 FD92' 30 026B 674 PUSHR #^M<R2,R4> ; Save registers needed after algpb.  
 B7 50 E9 026E 675 BSBW RMSALGBPBP ; Get Global Buffer Pointer Block.  
 FD8C' 30 0271 676 BLBC R0, ALGBPERR ; Branch on error.  
 B8 50 E9 0274 678 BSBW RM\$ALGBPBP ; Get Global Buffer Pointer Block.  
 FD86' 30 0277 679 BLBC R0, ALGBPERR1 ; Branch on error.  
 A4 50 E9 027A 680 BLBC R0, ALBLBERR1 ; Get a BLB.  
 FD80' 30 027D 681 BSBW RM\$ALBLB ; Branch on error.  
 97 50 E9 0280 682 BLBC R0, ALBLBERR ; Get a BLB.  
 14 BA 0283 683 POPR #^M<R2,R4> ; Exit on error.  
 0088 CA D5 0285 684 TSTL IFBSL\_GBH\_PTR(R10) ; Restore registers.  
 07 13 0289 685 BEQL 1\$ ; Already have gbl buffs?  
 FF04 31 028F 686 SSB #IRBV\$V\_GBLBUFF, (R9) ; No, then go on to map it.  
 0292 687 BRW EXIT ; Note irab has extra gpb, blb.  
 0292 688  
 0292 689 ; Branch to exit.  
 0292 690 ; R2 = Number of bytes to allocate (rounded up to full pages)  
 0292 691 ;  
 0292 692 ;  
 7E 7C 0292 693 1\$: CLRQ -(SP) ; Zero INADR forces P0 space to be allocated  
 7E 7C 0294 694 CLRQ -(SP) ; Reserve space for RETADR.  
 0296 695 ;  
 0296 696 ;  
 0296 697 ; The section name will be the ascii text '\_RMSS' followed by the  
 0296 698 ; FCB address in hexadecimal.  
 0296 699 ;  
 0296 700 ;  
 5E 10 C2 0296 701 SUBL2 #16, SP ; Make room for gsd name.  
 6E DF 0299 702 PUSHAL (SP) ; Addr part of descriptor.  
 0D DD 029B 703 PUSHL #13 ; Length of GSD name.  
 020C 8F BB 029D 704 PUSHR #^M<R2,R3,R9> ; Save these around GETCCB call.  
 59 5A D0 02A1 705 MOVL R10, R9 ; Need ifab in r9.  
 FD59' 30 02A4 706 BSBW RMSGETCCB ; Get CCB addr into R1.  
 020C 8F BA 02A7 707 POPR #^M<R2,R3,R9> ; Restore registers.  
 51 04 A1 D0 02AB 708 MOVL CCBSL\_WIND(R1), R1 ; Get ptr to window.  
 51 18 A1 D0 02AF 709 MOVL WCBSL\_FCB(R1), R1 ; Get FCB addr into R1.  
 7E FD4A CF DF 02B3 710 PUSHAL FAOCNTRL+1 ; Build descriptor for control string.  
 50 5E D0 02B7 711 MOVZBL FAOCNTRL, -(SP) ; Need to pass addr of desc.  
 7E FD45 CF 9A 02B7 712 MOVL SP, R0

6E	7C	02BF	713	\$FA0_S	CTRSTR=(R0) -	Address of control string descriptor		
		02BF	714		OUTBUF=8(R0), -	Addr of output buffer descriptor.		
		02BF	715		P1=R1	FCB addr to show up in output string.		
51	5E	DO	02CF	716	CLRQ (SP)	Clear priv mask.		
		02D1	717	SSB	#PRV\$V_SYSGBL, (SP)	Need sysgbl privilege.		
		02D5	718	MOVL	SP, R1	Save this stack address.		
		02D8	719	\$SETPRV_S	ENBFLG=#1,-	Turn on sysgbl for crmpsc.		
		02D8	720		PRVADR=(R1), -			
		02D8	721		PRVPRV=(R1)			
		02E7	722			Get previous state.		
50	51	08	AE	DE	02E7	723	MOVAL 8(SP), R1	Address of gsd name desc.
52	52	17	17	9C	02EB	724	ROTL #23, R2, R0	Get page count into r0.
					02EF	725	SCRMPSC_S INADR = 32(R1), -	Point to array on stack.
					02EF	726	RETADR = 24(R1), -	Point to array on stack.
					02EF	727	GSDNAM = (R1), -	"
					02EF	728	PAGCNT = R0, -	Number of pages in section.
					02EF	729	ACMODE = #P\$LSC EXEC, -	Access mode is EXEC.
					02EF	730	FLAGS = #SEC\$M_GBL!SEC\$M_SYSGBL!SEC\$M_WRT!SEC\$M_DZRO!SEC\$M_PAGFIL!SEC\$M_EXPR	
					0313	731		
1D	6E	19	E0	0313	732		BBS #PRV\$V_SYSGBL, (SP), 5\$	If already had sysgbl, skip turnoff.
	6E	7C	0317	733		CLRQ (SP)	Init priv mask.	
			0319	734		SSB #PRV\$V_SYSGBL, (SP)	Turn off sysgbl.	
51	5E	DO	031D	735		MOVL SP, R1	Address of priv mask.	
	50	DD	0320	736		PUSHL R0	Save status from crmpsc.	
	50	8E	DO	0322	737	\$SETPRV_S PRVADR=(R1)	Turn off sysgbl.	
			0331	738		POPL R0	Restore crmpsc status.	
	5E	20	C0	0334	739	5\$:	ADDL2 #32, SP	Clean priv mask+name desc +name.
	06	50	E8	0337	740		BLBS R0, 20\$	Continue if Ok.
	00ED	31	033A	741		BRW SEC_ERR	Branch to error code.	
	00E3	31	033D	742		BRW SEC_ERR1	Branch to error code.	
			0340	743	10\$:			
			0340	744	20\$:			
51	04	AE	6E	C3	0340	745	SUBL3 (SP), 4(SP), R1	Get size allocated - 1.
	51	D6	0345	746		INCL R1	Size allocated.	
52	51	D1	0347	747		CMPL R1, R2	Get everything?	
	F1	12	034A	748		BNEQ 10\$	Br if not.	
	53	6E	DO	034C	749	MOVL (SP), R3	Move starting address of section into R3.	
50	0619	8F	B1	034F	750	CMPW #SSS_CREATED, R0	Was the section just created?	
	0B	13	0354	751		BEQL 30\$	Then it needs to be initialized.	
08	A3	1611	8F	B1	0356	752	CMPW #<GBHSC_BID+<GBHSC_BLN/408>, GBHSC_BID(R3) ; Seem legit?	
		DF	12	035C	753	BNEQ 10\$	NEQ there's an error.	
		0087	31	035E	754	BRW STORE_PTR	Else use it.	
					0361	755		
					0361	756		
					0361	757		Initialize newly created section.
					0361	758		R3 = start address of section
					0361	759		R2 = size of section in bytes
					0361	760		R6 = number of buffers in section.
					0361	761		
					0361	762		
0C	A3	01	CE	0361	763	30\$:	MNEG L #1, GBHSL_HI_VBN(R3)	Store hi vbn for scan end check.
10	A3	52	DO	0365	764		MOVL R2, GBHSL_GS_SIZE(R3)	Store size of section in section.
38	A4	52	DO	0369	765		MOVL R2, GBSBSCE_G5_SIZE(R4)	Store size of section in GBSB.
34	A4	56	B0	036D	766		MOVW R6, GBSBSW_GBC(R4)	Store number of buffers in section.
08	A3	1611	8F	B0	0371	767	MOVW #<GBHSC_BID+<GBHSC_BLN/408>, GBHSC_BID(R3) ; Store id, bln.	
50	0058	8F	3C	0377	768		MOVZWL #GBHSC_BLN, R0	Offset to first GBD from GBH.
04	A3	50	DO	037C	769		MOVL R0, GBHSL_GBD_BLNK(R3)	Back link to GBD's.

28 A3 50 D0 0380 770 MOVL R0, GBHSL\_GBD\_START(R3) ; Save offset to first GBD.  
 30 A3 50 D0 0384 771 MOVL R0, GBHSL\_GBD\_NEXT(R3) ; First GBD is first victim.  
 34 A3 08 D0 0388 772 MOVL #8, GBHSL\_SCAN\_NUM(R3) ; Assume scan size of 8.  
 56 08 D1 038C 773 CMPL #8 R6 ; Have at least 8 buffers?  
 04 1B 038F 774 BLEQU 45\$ ; LEQU just use 8.  
 34 A3 56 D0 0391 775 MOVL R6, GBHSL\_SCAN\_NUM(R3) ; Else only use # in section.  
 56 D7 0395 776 45\$: DECL R6 ; Num - 1.  
 56 28 C4 0397 777 MULL2 #GBD\$C\_BLN, R6 ; Offset to last GBD from first.  
 52 01FF 8F AA 03A2 778 ADDL3 #GBD\$C\_BLN+GBHSC\_BLN+511, R6, R2 ; End of GBD's + page-1 byte.  
 50 53 C0 03A7 780 BICW2 #511, R2 ; Round off to even page.  
 56 50 C0 03AA 781 ADDL2 R3, R0 ; Start address of GBD's.  
 63 56 53 C3 03AD 782 ADDL2 R0, R6 ; Addr of last GBD.  
 2C A3 63 D0 03B1 784 ASSUME GBHSL\_GBD\_FLNK\_EQ 0 ; Forw Link points to last GBD.  
 04 A0 28 CE 03B5 785 50\$: SUBL3 R3, R6, (R3) ; Offset to last GBD.  
 60 28 D0 03B8 786 MOVL (R3), GBHSL\_GBD\_END(R3) ;  
 04 A0 28 CE 03B5 787 ASSUME GBD\$L\_FLINK\_EQ 0 ;  
 04 A0 28 D0 03B8 788 MNEGL #GBD\$C\_BLN, (R0) ; Offset to next GBD.  
 08 A0 0A13 8F B0 03BC 790 MOVL #GBD\$C\_BLN, GBD\$L\_BLINK(R0) ; Offset to last GBD.  
 0C A0 01 CE 03C2 791 ASSUME GBD\$B\_BLN\_EQ <GBD\$B\_BID + 1> ;  
 1A A0 55 B0 03C6 792 MOVW #<GBD\$C\_BID+<GBD\$C\_BLN/408>>, GBD\$B\_BID(R0) ; Id and bln.  
 1C A0 52 D0 03CA 793 MNEGL #1, GBD\$L\_VBN(R0) ; Init VBN to -1.  
 52 55 C0 03CE 794 MOVW R5, GBD\$W\_SIZE(R0) ; Store buffer size.  
 FFDE 50 28 56 F1 03D1 795 ADDL2 R2, GBD\$L\_REL\_ADDR(R0) ; Store offset to buffer.  
 04 A6 63 CE 03D7 796 ACBL R5, R2 ; Point to next buffer.  
 04 A3 CE 03DB 798 ASSUME R6, #GBD\$C\_BLN, R0, 50\$ ; Loop until past last GBD.  
 58 A3 CE 03DB 799 MNEGL (R3), GBD\$L\_BLINK(R6) ; Last GBD's back link is  
 04 A3 CE 03DE 800 MNEGL GBHSL\_GBD\_BLNK(R3), - ; opposite of header's forw link.  
 03E0 801 MNEGL GBHSC\_BLN+GBD\$L\_FLINK(R3) ; First GBD's forw link is  
 03E0 802 MNEGL GBHSL\_GBD\_BLNK(R3), - ; opposite of header's back link.  
 03E0 803 : If tracing is to be enabled, noop the following branch.  
 03E0 804 :  
 03E0 805 :  
 00000028'EF 06 11 03E0 806 BRB STORE\_PTR : To make it easy to patch in tracing.  
 17 03E2 807 JMP INIT\_TRC : To init tracing blocks.  
 0088 CA 5E 10 C0 03E8 808 STORE\_PTR: ADDL2 #16, SP ; 'Pop' INADR, RETADR arrays off stack.  
 1C A3 10 D6 03EB 809 INCL GBHSL\_USECNT(R3) ; Increment accessor count for section.  
 53 D0 03EE 810 MOVL R3, IFBSL\_GBH\_PTR(R10) ; Point to the section.  
 59 D5 03F3 811 TSTL R9 ; Irab present?  
 04 13 03F5 812 BEQL 20\$ ; EQL then no irab.  
 1C A3 01 D1 03FB 813 SSB #IRBV\_GBLBUFF, (R9) ; Note this irab has extra gpb, blb.  
 16 12 03FF 814 CMPL #1, GBHSL\_USECNT(R3) ; Are we first accessor?  
 0080 CA D0 0401 815 20\$: BNEQ 30\$ ; No, branch to release lock.  
 14 A3 0405 816 MOVL IFBSL\_PAR\_LOCK\_ID(R10), - ; Save file lock id in global section.  
 FBF6' 30 0407 817 BSBW RMSLOWER\_SYSLOCK ; Turn file lock into system lock.  
 FBF3' 30 040A 818 BSBW RMSLOWER\_GBS\_LOCK ; Lower lock on global buffer section.  
 54 78 AA D0 040D 819 MOVL IFBSL\_SF5B\_PTR(R10), R4 ; Put address of SF5B in R4 for INIT\_SF5B.  
 FBF3' 30 0411 820 BSBW RMSINIT\_SF5B\_IRB ; Get a file lock for process using IRB to s  
 FD7F' 31 0414 821 BRW EXIT ; Continue.  
 14 A3 D0 0417 822 MOVL GBHSL\_LOCK\_ID(R3), - ; Move the parent lock id for bucket  
 0080 CA 041A 823 BSBW IFBSL\_PAR\_LOCK\_ID(R10) ; locks into ifab from global buffer header  
 FBF0' 30 041D 824 30\$: BSBW RMSLOWER\_GBS\_LOCK ; Do lock mode conversion.

FD73 31 0420 827 BRW EXIT ; Continue.

0421 828

0422 829

0423 830 : An error has been detected. Disassociate from section, return structures

0423 831 : already allocated.

0423 832 :

0423 833

0423 834 SEC\_ERR1:

50 000184D4 BF DO 0423 835 MOVL #RMSS\_DME, R0 ; Give DME error if not all mapped.

042A 836 SEC\_ERR:

042A 837 SSUME FABSL STV EQ RABSL\_STV

OC A8 50 DO 042A 838 MOVL RABSL\_STV(R8)

50 8E 7D 042E 839 MOVQ (SP)+, R0- ; Save error code.

5E 08 C0 0431 840 ADDL2 #8, SP ; Get RETADR off stack into r0 and r1.

FBC4' 30 0439 841 RMSERR CRMP, -(SP) ; Pop INADR off stack.

FDAD 31 043C 842 BSBW RMSUNMAP\_GBL\_ALT ; Note error.

043F 843 BRW ERLO ; Delete the whole VA.

844 ; Branch to finish up.

00000190 043F 846 \$NEWPSECT RM\$TRACE  
0000 847 NUMTRC: .LONG 400 ; Number of trace blocks to allocate.  
0004 848  
0004 849 ;  
0004 850 ; Add in extra bytes for trace blocks after size of section is determined.  
0004 851 ;  
0004 852 ;  
0004 853 ADDTRC:  
50 F9 AF D0 0004 854 MOVL NUMTRC, R0 ; Get number of trace blocks desired.  
02 12 0008 855 BNEQ 10\$ Branch if non-zero.  
50 D6 000A 856 INCL R0 ; Get at least one.  
50 00000040 8F C4 000C 857 10\$: MULL2 #TRCSC\_BLN, R0 ; Get size of trace blocks.  
000001FF 8F C0 0013 858 ADDL2 #511, R0 ; Add in almost a page.  
50 01FF 8F AA 001A 859 BICW2 #511, R0 ; Round to even page's worth.  
52 50 C0 001F 860 ADDL2 R0, R2 ; Add in to size being requested.  
00000269'EF 17 0022 861 JMP MAP\_IT ; And return to mainline.  
0028 862  
0028 863 ; Initialize the trace blocks and pointer from the global buffer header.  
0028 864 ;  
0028 865 ; R3 - pointer to GBH  
0028 866 ; R5 - buffer size  
0028 867 ;  
0028 868 ;  
0028 869 ;  
0028 870 INIT\_TRC:  
50 53 63 C1 0028 871 ASSUME GBH\$L\_GBD\_FLNK EQ 0  
55 1C A0 C1 002C 872 ADDL3 (R3), R3, R0 ; Get address of last GBD in list.  
20 A3 50 20 C3 0031 873 ADDL3 GBD\$L\_REL\_ADDR(R0), R5, R0 ; R0 now first byte after last buff.  
50 53 C0 0036 874 SUBL3 #GBH\$L\_TRC\_FLNK, R0, GBH\$L TRC\_FLNK(R3) ; Offset to 1st trc blk  
51 53 10 A3 C1 0039 875 ADDL2 R3, R0 ; R0 now addr of first trace block.  
51 00000040 8F C2 003E 876 ADDL3 GBH\$L GS\_SIZE(R3), R3, R1 ; Get addr of end of gbl sec.  
0045 877 SUBL2 #TRCSC\_BLN, R1 ; Limit for last trace block.  
10\$: 0045 878 ASSUME <TRCSC\_BLN & 7> EQ 0 ; These will line up on quad boundary.  
0045 879 ASSUME TRCSL\_FLNK EQ 0  
80 00000040 8F D0 0045 880 MOVL #TRCSC\_BLN, (R0)+ ; Fwd offset to next block.  
004C 881 ASSUME TRCSL\_BLNK EQ 4  
80 00000040 8F CE 004C 882 MNEGL #TRCSC\_BLN, (R0)+ ; Back offset to last block.  
0053 883 ASSUME TRCSB\_BID EQ 8  
80 1012 8F B0 0053 884 ASSUME TRCSB\_BLN EQ <TRCSB\_BID + 1>  
FFE7 50 36 51 F1 0058 885 MOVW #<TRCSC\_BID+<TRCSC\_BLN/4a8>, (R0)+ ; Store id and bln.  
005E 886 ACBL R1, #TRCSC\_BLN-10, R0, 10\$ ; Keep going until past limit.  
50 00000040 8F C2 005E 887  
51 53 C3 0065 888 SUBL2 #TRCSC\_BLN, R0 ; Back up to last trace block.  
24 A3 51 20 C3 0069 889 SUBL3 R3, R0, R1 ; R1 is offset to last trc blk.  
60 24 A3 CE 006E 890 SUBL3 #GBH\$L TRC\_FLNK, R1, GBH\$L TRC\_BLNK(R3) ; Back link in header.  
50 20 A3 DE 0072 891 MNEGL GBH\$L TRC\_BLNK(R3).TRCSL\_FLNK(R0) ; Flnk to hdr from last trc.  
50 60 C0 0076 892 MOVAL GBH\$L TRC\_FLNK(R3), R0 ; Addr of flnk from header.  
04 A0 20 A3 CE 0079 893 ADDL2 (R0), R0 ; Get first trace block.  
000003E8'EF 17 007E 894 MNEGL GBH\$L TRC\_FLNK(R3), TRCSL\_BLNK(R0) ; Fix it's back link.  
0084 895 JMP STORE\_PTR ; Jump back to main line.  
0084 896  
0084 897

0084	899	;			
0084	900	:	Routine called to store information in trace block from initial call		
0084	901	:	to cache routine.		
0084	902	:			
0084	903	:	AP is destroyed. All other registers preserved.		
0084	904	:			
0084	905	RMSCACHE_IN::			
50 0088 03 CA	BB 0084	906	PUSHR #^M<R0,R1>	;	Save registers used.
44	D0 0086	907	MOVL IFBSL_GBH_PTR(R10), R0	:	Get pointer to gbh, if any.
0236	13 008B	908	BEQL EX2	:	Exit if none.
3F	30 008D	909	BSBW REMQT	:	Get a trace block.
50 0A A0	9E 0092	910	BEQL EX2	:	Exit if none.
80 01	B0 0096	911	MOVAB TRCSW_FUNCTION(R0), R0	:	Get addr of function cell.
80 59	D0 0099	912	MOVW #GBHSM_CACHE_IN, (R0)+	:	Note this function.
5C 00000000'9F	D0 009C	913	MOVL R9, (R0)+	:	structure
80 60 AC	B0 00A3	914	MOVL @#CTL\$GL_PCB, AP	:	Get pcb addr.
0219	30 00A7	915	MOVW PCB\$L_PID(AP), (R0)+	:	pid
80 04 AE	D0 00AA	916	BSBW CNT	:	seqnum
80 0C AE	D0 00AE	917	MOVL 4(SP), (R0)+	:	vbn
80 20 AE	D0 00B2	918	MOVL 12(SP), (R0)+	:	return1
80 53	D0 00B6	919	MOVL 32(SP), (R0)+	:	return2
80	D4 00B9	920	MOVL R3, (R0)+	:	arg_flg
80	7C 00BB	921	CLRL (R0)+	:	bdb_addr
80	7C 00BD	922	CLRQ (R0)+	:	not used
80	7C 00BF	923	CLRQ (R0)+	:	not used
80	7C 00BF	924	CLRQ (R0)+	:	not used
51 50 00000040 8F	C3 00C1	925	SUBL3 #TRCS_C_BLN, R0, R1	:	Get addr of trc blk
50 0088 CA	D0 00C9	926	MOVL IFBSL_GBH_PTR(R10), R0	:	Get addr of gbh.
01FF	30 00CE	927	BSBW INSQH	:	Insert blk at head of list.
03	BA 00D1	928	EX2: POPR #^M<R0,R1>	:	Restore registers.
05	00D3	929	RSB	:	Return to cache
	00D4	930			
	00D4	931			

		00D4	933	;				
		00D4	934	;	Store useful information from cache exit.			
		00D4	935	;				
		00D4	936	;				
		00D4	937	RM\$CACH_OUT::				
50	0088	03	BB	00D4	938	PUSHR #^M<R0,R1>	Save registers.	
		CA	D0	00D6	939	1\$: MOVL IFBSL_GBH_PTR(R10), R0	Get GBH ptr, if any.	
		F4	13	00DB	940	BEQL EX2	Exit if none.	
		01E6	30	00DD	941	BSBW REMQT	Remove a trc blk from tail.	
		EF	13	00E0	942	BEQL EX2	Exit if none.	
50	0A	A0	9E	00E2	943	MOVAB TRCSW_FUNCTION(R0), R0	Ptr to func field.	
	80	02	B0	00E6	944	MOVW #GBHSM_CACHE_OUT, (R0)+	function	
	80	59	DO	00E9	945	MOVL R9, (R0)+	structure	
5C	00000000	'9F	DO	00EC	946	MOVL @#CTL\$GL_PCB, AP	Addr of PCB	
	80	60	AC	B0	947	MOVL PCBSL_PID(AP), (R0)+	pid	
		01C9	30	00F7	948	BSBW CNT	structure	
		80	D4	00FA	949	CLRL (R0)+	vbn	
80	0C	AE	DO	00FC	950	MOVL 12(SP), (R0)+	return1	
	80	24	AE	DO	951	MOVL 36(SP), (R0)+	return2	
	80	6E	DO	0100	952	MOVL (SP), (R0)+	arg_flg	
	80	54	DO	0107	953	MOVL R4, (R0)+	bdb_addr	
		45	13	010A	954	BEQL 10\$		
EC	A0	1C	A4	DO	010C	MOVL BDBSL_VBN(R4), -20(R0)		
	80	0C	A4	B0	0111	MOVW BDBSW_USERS(R4), (R0)+		
	80	0E	A4	B0	0115	MOVW BDBSW_BUFF_ID(R4), (R0)+		
	80	0B	A4	90	0119	MOVB BDBSB_CACHE_VAL(R4), (R0)+		
	80	0A	A4	90	011D	MOVB BDBSB_FLGS(R4), (R0)+		
	80	20	A4	DO	0121	MOVL BDBSL_VBNSEQNO(R4), (R0)+		
51	10	A4	DO	0125	961	MOVL BDBSL_BLB_PTR(R4), R1		
		2E	13	0129	962	BEQL 20\$		
	80	0B	A1	90	012B	MOVB BLBSB_MODEHELD(R1), (R0)+		
	80	0A	A1	90	012F	MOVB BLBSB_BLBFLGS(R1), (R0)+		
	80	51	DO	0133	965	MOVL R1, (R0)+		
	80	24	A1	DO	0136	MOVL BLBSL_LOCK_ID(R1), (R0)+		
	80	28	A1	DO	013A	MOVL BLBSL_VALSEQNO(R1), (R0)+		
				013E	968	5\$: SUBL3 #TRCSL_BLN, R0, R1	Get ptr to trc blk to insert.	
51	50	00000040	8F	C3	013E	969	MOVL IFBSL_GBH_PTR(R10), R0	Insert at head of queue.
	50	0088	CA	D0	0146	970	BSBW INSQH	Branch to exit.
		0182	30	014B	971	BRW EX2		
		FF80	31	014E	972	CLRQ (R0)+		
		80	7C	0151	973	10\$: CLRQ (R0)+		
		80	7C	0153	974	CLRQ (R0)+		
		80	7C	0155	975	15\$: CLRQ (R0)+		
		E5	11	0157	976	BRB 5\$		
		80	B4	0159	977	20\$: CLRW (R0)+		
		80	D4	015B	978	CLRL (R0)+		
		F6	11	015D	979	BRB 15\$		

015F	981				
015F	982	;			
015F	983	;	Store trace info for initial call to release.		
015F	984	;			
015F	985				
015F	986	RMSRLS_IN::			
50 0088	07 CA	BB 015F	987	PUSHR #^M<R0,R1,R2>	
	03	DO 0161	988	1\$: MOVL IFBSL_GBH_PTR(R10), R0	
	00A3	12 0166	989	BNEQ 3\$	
	0158	31 0168	990	BRW EX1	
	03	30 016B	991	BSBW REMQT	: Get trc blk from end.
	03	12 016E	992	BNEQ 4\$	Branch if got one.
	009B	31 0170	993	BRW EX1	Else exit.
50	0A A0	9E 0173	994	4\$: MOVAB TRCSW FUNCTION(R0), R0	
	80 04	BO 0177	995	MOVW #GBH\$M RLS_IN, (R0)+	: function
	80 59	DO 017A	996	MOVL R9, (R0)+	: structure
5C	00000000 9F	DO 017D	997	MOVL @#CTL_SGL_PCB, AP	
	80 60 AC	BO 0184	998	MOVW PCB\$L_PID(AP), (R0)+	: pid
	0138	30 0188	999	BSBW CNT	: seqnum
		018B	1000		
	51	7C 018B	1001	CLRQ R1	
	54	D5 018D	1002	TSTL R4	
	04	12 018F	1003	BNEQ 5\$	
	80	D4 0191	1004		
	20	11 0193	1005	CLRL (R0)+	: VBN
		0195	1006	BRB 50\$	
	55:	0195	1007		
08 A4	10	91 0195	1008	CMPB #BLB\$C_BID, BLB\$B_BID(R4)	
	05	12 0199	1009	BNEQ 20\$	
51	54	DO 019B	1010	MOVL R4, R1	
51	07	11 019E	1011	BRB 30\$	
51	10 A4	DO 01A0	1012	20\$: MOVL BDB\$L_BLB_PTR(R4), R1	
52	54	DO 01A4	1013	MOVL R4, R2	
52	52	D5 01A7	1014	30\$: TSTL R2	: IS THERE BDB?
	06	13 01A9	1015	BEQL 40\$	
80	1C A2	DO 01AB	1016	MOVL BDB\$L_VBN(R2), (R0)+	
	04	11 01AF	1017	BRB 50\$	
80	14 A1	DO 01B1	1018	40\$: MOVL BLB\$L_VBN(R1), (R0)+	
80	10 AE	DO 01B5	1019	50\$: MOVL 16(SP), (R0)+	: RETURN1
80	20 AE	DO 01B9	1020	MOVL 32(SP), (R0)+	: RETURN2
80	53	DO 01BD	1021	MOVL R3, (R0)+	: FLAGS
80	52	DO 01C0	1022	MOVL R2, (R0)+	: BDB ADDR
	16	13 01C3	1023	BEQL 60\$	
80	0C A2	BO 01C5	1024	MOVW BDB\$W_USERS(R2), (R0)+	
80	0E A2	BO 01C9	1025	MOVW BDB\$W_BUFF_ID(R2), (R0)+	
80	0B A2	90 01CD	1026	MOVB BDB\$B_CACHE_VAL(R2), (R0)+	
80	0A A2	90 01D1	1027	MOVB BDB\$B_FLGS(R2), (R0)+	
80	20 A2	DO 01D5	1028	MOVL BDB\$L_VBNSEQNO(R2), (R0)+	
	04	11 01D9	1029	BRB 70\$	
	01DB	01DB	1030	60\$:	
80	7C	01DB	1031	CLRQ (R0)+	
80	B4	01DD	1032	CLRW (R0)+	
51	D5	01DF	1033	70\$: TSTL R1	: IS THERE BLB?
	15	13 01E1	1034	BEQL 80\$	
80	0B A1	90 01E3	1035	MOVW BLB\$B_MODEHELD(R1), (R0)+	
80	0A A1	90 01E7	1036	MOVW BLB\$B_BLBFLGS(R1), (R0)+	
80	51	DO 01EB	1037	MOVL R1, (R0)+	

80 24 A1 D0 01EE 1038 MOVL BLBSL\_LOCK\_ID(R1), (R0)+  
80 28 A1 D0 01F2 1039 MOVL BLBSL\_VALSEQNO(R1), (R0)+  
06 11 01F6 1040 BRB 90\$:  
80 B4 01F8 1041 80\$: CLRW (R0)+  
80 D4 01FA 1042 CLRL (R0)+  
80 7C 01FC 1044 CLRQ (R0)+  
51 50 00000040 8F C3 01FE 1045 90\$: SUBL3 #TRC\$C\_BLN, R0, R1  
50 0088 CA D0 0206 1046 MOVL IFBSL\_GBH\_PTR(R10), R0  
00C2 30 020B 1047 BSBW INSQH ; Insert element at head of queue.  
07 BA 020E 1048 EX1: POPR #^M<R0,R1,R2>  
05 0210 1050 RSB

```

0211 1052
0211 1053
0211 1054 ; Store trace info at exit of release routine.
0211 1055
0211 1056
0211 1057 RM$RLS_OUT:::
50 0088 07 BB 0211 1058 PUSHR #^M<R0,R1,R2>
      CA D0 0213 1059 1$: MOVL IFBSL_GBH_PTR(R10), R0
      03 12 0218 1060 BNEQ 3$:
      FFF1 31 021A 1061 BRW EX1
      00A6 30 021D 1062 3$: BSBW REMQT
      03 12 0220 1063 BNEQ 4$:
      FFE9 31 0222 1064 BRW EX1
      50 0A A0 9E 0225 1065 4$: MOVAB TRCSW_FUNCTION(R0), R0
      80 08 B0 0229 1066 MOVW #GBHSM_RLS_OUT, (R0)+ ; function
      80 59 D0 022C 1067 MOVL R9, (R0)+ ; structure
      SC 00000000'9F D0 022F 1068 MOVL a#CTL$GL_PCB, AP
      80 60 AC B0 0236 1069 MOVW PCB$L_PID(AP), (R0)+ ; pid
      0086 30 023A 1070 BSBW CNT ; seqnum

      51 7C 023D 1071 CLRQ R1
      54 D5 023F 1073 TSTL R4
      04 12 0241 1074 BNEQ 5$:
      0243 1075
      80 D4 0243 1076 CLRQ (R0)+ ; VBN
      20 11 0245 1077 BRB 50$:
      0247 1078 5$: CMPB #BLB$C_BID, BLB$B_BID(R4)
      08 A4 10 91 0247 1079 BNEQ 20$:
      51 54 D0 024B 1080 MOVL R4, R1
      51 07 11 0250 1082 BRB 30$:
      51 10 A4 D0 0252 1083 20$: MOVL BDB$L_BLB_PTR(R4), R1
      52 54 D0 0256 1084 MOVL R4, R2 ; IS THERE BDB?
      52 D5 0259 1085 30$: TSTL R2
      06 13 025B 1086 BEQL 40$:
      80 1C A2 D0 025D 1087 MOVL BDB$L_VBN(R2), (R0)+ ; RETURN1
      04 11 0261 1088 BRB 50$:
      80 14 A1 D0 0263 1089 40$: MOVL BLB$L_VBN(R1), (R0)+ ; RETURN2
      80 10 AE D0 0267 1090 50$: MOVL 16(SP), (R0)+ ; STATUS
      80 20 AE D0 026B 1091 MOVL 32(SP), (R0)+ ; BDB ADDR
      80 6E D0 026F 1092 MOVL (SP), (R0)+ ; BDB$W_USERS(R2), (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      80 52 D0 0272 1093 BEQL 60$:
      80 16 13 0275 1094 MOVL (R0)+ ; BDB$W_USERS(R2), (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      80 0C A2 B0 0277 1095 MOVL (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      80 0E A2 B0 027B 1096 MOVL (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      80 0B A2 90 027F 1097 MOVL (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      80 0A A2 90 0283 1098 MOVL (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      80 20 A2 D0 0287 1099 MOVL (R0)+ ; BDB$W_BUFF_ID(R2), (R0)+ ; BDB$B_CACHE_VAL(R2), (R0)+ ; BDB$B_FLGS(R2), (R0)+ ; BDB$L_VBNSEQNO(R2), (R0)+ ; BDB$W
      04 11 028B 1100 BRB 70$:
      80 7C 028D 1101 60$: CLRQ (R0)+ ; IS THERE BLB?
      80 B4 028F 1103 CLRW (R0)+ ; IS THERE BLB?
      51 D5 0291 1104 70$: TSTL R1
      15 13 0293 1105 BEQL 80$:
      80 0B A1 90 0295 1106 MOVB BLB$B_MODEHELD(R1), (R0)+ ; BDB$W
      80 0A A1 90 0299 1107 MOVB BLB$B_BLBFLGS(R1), (R0)+ ; BDB$W
      80 51 D0 029D 1108 MOVL R1, (R0)+ ; BDB$W

```

80 24 A1 D0 02A0 1109  
80 28 A1 D0 02A4 1110  
06 11 02A8 1111  
02AA 1112 80\$: MOVL BLB\$L\_LOCK\_ID(R1), (R0)+  
80 B4 02AA 1113 MOVL BLB\$L\_VALSEQNO(R1), (R0)+  
50 D4 02AC 1114 BRB 90\$  
80 7C 02AE 1115 CLRW (R0)+  
02B0 1116 90\$: CLRL (R0)+  
CLRQ (R0)+  
51 50 00000040 8F C3 02B0 1117 SUBL3 #TRC\$C\_BLN, R0, R1  
50 0088 CA D0 02B8 1118 MOVL IFBSL\_GBH\_PTR(R10), R0  
0010 30 02BD 1119 BSBW INSQH ; Insert at head of queue.  
FF4B 31 02C0 1120 BRW EX1  
02C3 1121  
02C3 1122 CNT: CLRW (R0)+  
80 B4 02C3 1123 RSB  
05 02C5 1124  
02C6 1125  
02C6 1126 ;CRASH: RMSPBUG -99

02C6 1128 ;  
02C6 1129 ; Routine to remove an element from the end of a self relative queue.  
02C6 1130 ; The forward and back links in the removed element remain intact.  
02C6 1131 ;  
02C6 1132 ; Input: R0 - GBH header.  
02C6 1133 ; Output: R0 - trc blk element to use.  
02C6 1134 ; R1 destroyed.  
02C6 1135 ;  
02C6 1136 ;  
02C6 1137 REMQT:  
20 A0 D5 02C6 1138 TSTL GBH\$L\_TRC\_FLNK(R0) ; Make sure trace blocks exists.  
04 13 02C9 1139 BEQL 10\$ ; EQL there aren't any.  
50 20 A0 5F 02CB 1140 REMQT1 GBH\$L\_TRC\_FLNK(R0),R0 ; Remove a trc block from end of queue.  
05 02CF 1141 10\$: RSB ; Return.  
02D0 1142 ;  
02D0 1143 ;  
02D0 1144 ; Routine to insert the trc blk previously removed from the tail of the queue  
onto the head of the queue.  
02D0 1145 ;  
02D0 1146 ;  
02D0 1147 ; Input:  
02D0 1148 ; R0 - GBH ptr.  
02D0 1149 ; R1 - element to insert.  
02D0 1150 ;  
02D0 1151 ;  
02D0 1152 INSQH:  
20 A0 61 5C 02D0 1153 INSQHI (R1),GBH\$L\_TRC\_FLNK(R0) ; Insert onto front of queue.  
05 02D4 1154 RSB ; And return.  
02D5 1155 ;  
02D5 1156 ; SPSECT\_RESTORE  
043F 1157 .END

\$.PSECT EP	= 00000000		FAB\$M_BIO	= 00000020	
SSRMSTEST	= 0000001A		FAB\$V_BRO	= 00000006	
SSRMS_PBUGCHK	= 00000010		FAB\$W_GBC	= 00000048	
SSRMS_TBUGCHK	= 00000008		FAOCNTRL	= 00000000 R 01	
SSRMS_UMODE	= 00000004		GBD\$B_BID	= 00000008	
SST1	= 00000000		GBD\$B_BLN	= 00000009	
SST2	= 00000004		GBD\$C_BID	= 00000013	
ADDTRC	00000004 R 03		GBD\$C_BLN	= 00000028	
AGAIN	00000169 R 01		GBD\$L_BLINK	= 00000004	
ALBLBERR	0000021A R 01		GBD\$L_FLINK	= 00000000	
ALBLBERR1	00000221 R 01		GBD\$L_REL_ADDR	= 0000001C	
ALGPBPERR	00000228 R 01		GBD\$L_VBN	= 0000000C	
ALGPBPERR1	0000022F R 01		GBD\$W_SIZE	= 0000001A	
ALLOC	0000009D R 01		GBH\$B_BID	= 00000008	
BAD_GBC	00000236 R 01		GBH\$C_BID	= 00000011	
BDB\$B_CACHE_VAL	= 00000008		GBH\$C_BLN	= 00000058	
BDB\$B_FLGS	= 0000000A		GBH\$L_GBD_BLNK	= 00000004	
BDB\$L_BLB_PTR	= 00000010		GBH\$L_GBD_END	= 0000002C	
BDB\$L_VBN	= 0000001C		GBH\$L_GBD_FLINK	= 00000000	
BDB\$L_VBNSEQNO	= 00000020		GBH\$L_GBD_NEXT	= 00000030	
BDB\$W_BUFF_ID	= 0000000E		GBH\$L_GBD_START	= 00000028	
BDB\$W_USERS	= 0000000C		GBH\$L_GS_SIZE	= 00000010	
BLB\$B_BID	= 00000008		GBH\$L_HI_VBN	= 0000000C	
BLB\$B_BLBFLGS	= 0000000A		GBH\$L_LOCK_ID	= 00000014	
BLB\$B_MODEHELD	= 0000000B		GBH\$L_SCAN_NUM	= 00000034	
BLB\$C_BID	= 00000010		GBH\$L_TRC_BLNK	= 00000024	
BLB\$L_LOCK_ID	= 00000024		GBH\$L_TRC_FLINK	= 00000020	
BLB\$L_VALSEQNO	= 00000028		GBH\$L_USECNT	= 0000001C	
BLB\$L_VBN	= 00000014		GBH\$M_CACHE_IN	= 00000001	
BLKALC	0000010D R 01		GBH\$M_CACHE_OUT	= 00000002	
CCBSL_WIND	= 00000004		GBH\$M_RLS_IN	= 00000004	
CHKGBC	000001B6 R 01		GBH\$M_RLS_OUT	= 00000008	
CHKMBC	0000004A R 01		GBSB\$C_GS_SIZE	= 00000038	
CHK_GBC	0000023D R 01		GBSB\$W_GBC	= 00000034	
CLNT	00000036 R 01		GIVEBACK	000001A9 R 01	
CNT	000002C3 R 03		IFB\$B_FAC	= 00000022	
CTL\$GL_PCB	***** X 03		IFB\$L_BDB_BLNK	= 00000044	
DECR_BCNT	000000194 R 01		IFB\$L_BLBBLNK	= 0000009C	
DEV\$V_FOD	= 0000000E		IFB\$L_DEVBUFSIZ	= 00000048	
DEV\$V_FOR	= 00000018		IFB\$L_EBK	= 00000074	
DEV\$V_REC	= 00000000		IFB\$L_GBH_PTR	= 00000088	
DEV\$V_RND	= 0000001C		IFB\$L_GBSB_PTR	= 0000007C	
DEV\$V_RTM	= 0000001D		IFB\$L_PAR_CLOCK_ID	= 00000080	
DEV\$V_SQD	= 00000005		IFB\$L_PRIM_DEV	= 00000000	
DEV\$V_TRM	= 00000002		IFB\$L_SFSB_PTR	= 00000078	
ERL0	000001EC R 01		IFB\$V_BIO	= 00000005	
ERL1	000001F4 R 01		IFB\$V_DAP	= 0000003E	
ERL2	0000020A R 01		IFB\$V_EOF	= 00000021	
ERL3	00000203 R 01		IFB\$V_NORECLK	= 00000033	
ERL4	000001FC R 01		IFB\$V_PPF_INPUT	= 0000002E	
ERRDEV	0000003C R 01		IFB\$V_TEF	= 00000036	
ERRMBC	00000043 R 01		IFB\$W_AVLCL	= 00000084	
ERRRFM	00000031 R 01		IFB\$W_FFB	= 0000005C	
EX1	0000020E R 03		IMP\$V_IIOS	= 00000000	
EX2	000000D1 R 03		IMP\$V_NOPBUFS	= 00000005	
EXIT	00000196 R 01		INIT_TRC	00000028 R 03	
FABSL_STV	= 0000000C		INSQR	000002D0 R 03	

IRBSB_BCNT	= 00000054	RMSS_MBC	= 00018734
IRBSB_MBC	= 00000055	RMSS_RFM	= 00018664
IRBSB_MBF	= 0000005C	ROP	= 00000020
IRBSL_NRP_VBN	= 00000040	SECSM_DZRO	= 00000004
IRBSL_NXTBDB	= 0000003C	SECSM_EXPREG	= 00020000
IRBSV_EOF	= 00000021	SECSM_GBL	= 00000001
IRBSV_GBLBUFF	= 00000036	SECSM_PAGFIL	= 00080000
IRBSV_RAHWBH	= 0000002A	SECSM_SYSGBL	= 00080000
IRBSW_NRP_OFF	= 00000044	SECSM_WRT	= 00000008
MAP_IT	00000269 R 01	SEC_ERR	0000042A R 01
NT\$CONNECT	***** X 01	SEC_ERR1	00000423 R 01
NUMTRC	00000000 R 03	SETNXT	000000BE R 01
PCBSL_PID	= 00000060	SS\$ CREATED	= 00000619
PIO\$GB_DFMBC	***** X 01	STORE PTR	000003E8 R 01
PIO\$GB_DFMBFSDK	***** X 01	SYSSCRMPSC	***** GX 01
PRV\$V_SYSGBL	= 00000019	SYSSFAO	***** X 01
PSL\$C_EXEC	= 00000001	SYSSGB_DFMBC	***** X 01
RABSB_MBC	= 00000037	SYSSGB_DFMBFSDK	***** X 01
RABSB_MBF	= 00000036	SYSSSETPRV	***** GX 01
RABSB_ROP1	= 00000005	TRCSB_BID	= 00000008
RABSL_FAB	= 0000003C	TRCSB_BLN	= 00000009
RABSL_ROP	= 00000004	TRCS\$C_BID	= 00000012
RABSL_STV	= 0000000C	TRCS\$C_BLN	= 00000040
RABSM_RAH	= 00000200	TRCSL_BLNK	= 00000004
RABSM_WBH	= 00000400	TRCSL_FLNK	= 00000000
RAB\$V_BIO	= 0000000B	TRCSW_FUNCTION	= 0000000A
RAB\$V_EOF	= 00000008	UNIT	000000A6 R 01
REMQT	000002C6 R 03	WCBSL_FCB	= 00000018
RMSALBLB	***** X 01		
RMSALDBUF	***** X 01		
RMSALGBP	***** X 01		
RMSBDBALLOC	00000111 RG 01		
RMSBDBALLOC_ALT	0000015E RG 01		
RMS\$CACH_IN	00000084 RG 03		
RMS\$CACH_OUT	000000D4 RG 03		
RMS\$CLNT	***** X 01		
RMS\$COMCLNUP	***** X 01		
RMS\$CONNECT1	00000009 RG 01		
RMS\$CONN_ERRRFM	00000031 RG 01		
RMS\$EXSUC	***** X 01		
RMS\$EX_NOSTR	***** X 01		
RMS\$GETCCB	***** X 01		
RMS\$INIT_GBSB	***** X 01		
RMS\$INIT_SFSB_IRB	***** X 01		
RMS\$LOWER_GBS_LOCK	***** X 01		
RMS\$LOWER_SYSLOCK	***** X 01		
RMS\$RETBDB	***** X 01		
RMS\$RETBBLB	***** X 01		
RMS\$RETGBPB	***** X 01		
RMS\$RLS_GBSB	***** X 01		
RMS\$RLS_IN	0000015F RG 03		
RMS\$RLS_OUT	00000211 RG 03		
RMS\$UNMAP_GBL_ALT	***** X 01		
RMSS_CRMP	= 0001C14C		
RMSS_DEV	= 000184C4		
RMSS_DME	= 000184D4		
RMSS_GBC	= 000187CC		

```
+-----+
! Psect synopsis !
+-----+
```

## PSECT name

PSECT name	Allocation	PSECT No.	Attributes
ABS	000000000 ( 0.)	00 ( 0.)	NOPIE USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM\$RMS1	0000043F ( 1087.)	C1 ( 1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
SABSS	000000000 ( 0.)	02 ( 2.)	NOPIE USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RM\$TRACE	000002D5 ( 725.)	03 ( 3.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE

```
+-----+
! Performance indicators !
+-----+
```

## Phase

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.08	00:00:00.40
Command processing	137	00:00:00.68	00:00:04.36
Pass 1	608	00:00:26.33	00:01:09.37
Symbol table sort	0	00:00:03.90	00:00:04.89
Pass 2	219	00:00:05.36	00:00:13.14
Symbol table output	32	00:00:00.22	00:00:00.74
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1036	00:00:36.60	00:01:32.94

The working set limit was 1950 pages.

145709 bytes (285 pages) of virtual memory were used to buffer the intermediate code.

There were 140 pages of symbol table space allocated to hold 2617 non-local and 71 local symbols.

1157 source lines were read in Pass 1, producing 20 object records in Pass 2.

45 pages of virtual memory were used to define 43 macros.

```
+-----+
! Macro library statistics !
+-----+
```

## Macro library name

Macro library name	Macros defined
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	19
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	4
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	16
TOTALS (all libraries)	39

2809 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LI\$:\$:RM1CONN/OBJ=OBJ\$:\$:RM1CONN MSRC\$:\$:RM1CONN/UPDATE=(ENH\$:\$:RM1CONN)+EXECMLS/LIB+LIB\$:\$:RMS/LIB

0321 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

RMICONN  
LIS

RMIGET  
LIS

RMIINPSN  
LIS

RMLDISCON  
LIS

RMIGETINT  
LIS

RMICREATE  
LIS

RMIJOURNL  
LIS